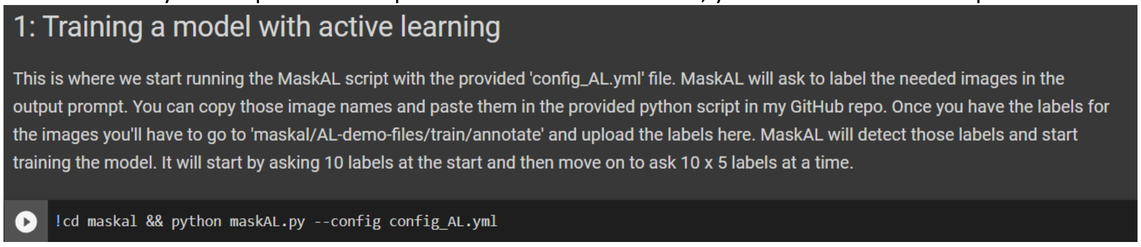
howest
hogeschool

User manual

Daan De Wilde

Running the active learning script

1. Once you completed all steps in the installation manual, you can run the first script.

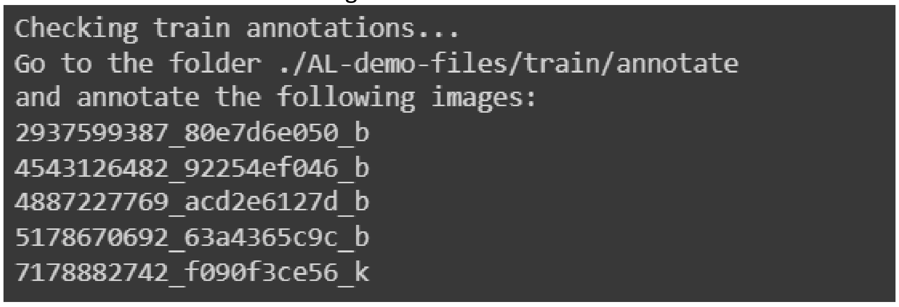


1: Training a model with active learning

This is where we start running the MaskAL script with the provided 'config_AL.yml' file. MaskAL will ask to label the needed images in the output prompt. You can copy those image names and paste them in the provided python script in my GitHub repo. Once you have the labels for the images you'll have to go to 'maskal/AL-demo-files/train/annotate' and upload the labels here. MaskAL will detect those labels and start training the model. It will start by asking 10 labels at the start and then move on to ask 10 x 5 labels at a time.

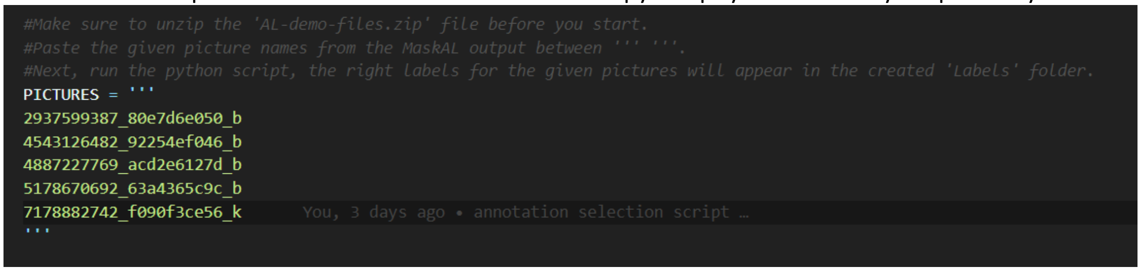
```
!cd maskal && python maskAL.py --config config_AL.yml
```

2. The MaskAL script will give you a list of images that need to be labelled. The very first time it will ask to label 10 images.



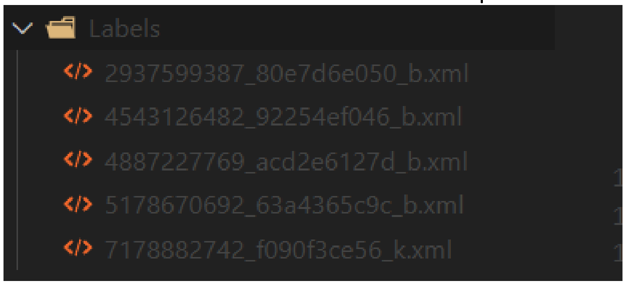
```
Checking train annotations...
Go to the folder ./AL-demo-files/train/annotate
and annotate the following images:
2937599387_80e7d6e050_b
4543126482_92254ef046_b
4887227769_acd2e6127d_b
5178670692_63a4365c9c_b
7178882742_f090f3ce56_k
```

3. You can paste this list in the 'select-annotations.py' script you cloned to your pc locally earlier.



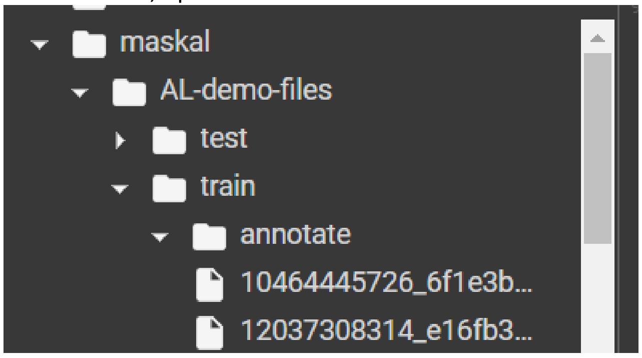
```
#Make sure to unzip the 'AL-demo-files.zip' file before you start.
#Paste the given picture names from the MaskAL output between ''' '''.
#Next, run the python script, the right labels for the given pictures will appear in the created 'Labels' folder.
PICTURES = '''
2937599387_80e7d6e050_b
4543126482_92254ef046_b
4887227769_acd2e6127d_b
5178670692_63a4365c9c_b
7178882742_f090f3ce56_k        You, 3 days ago • annotation selection script …
'''
```

4. Once the image names are pasted between the "'   '", you can run the python script. It will create a 'Labels' folder with the requested labels in it.



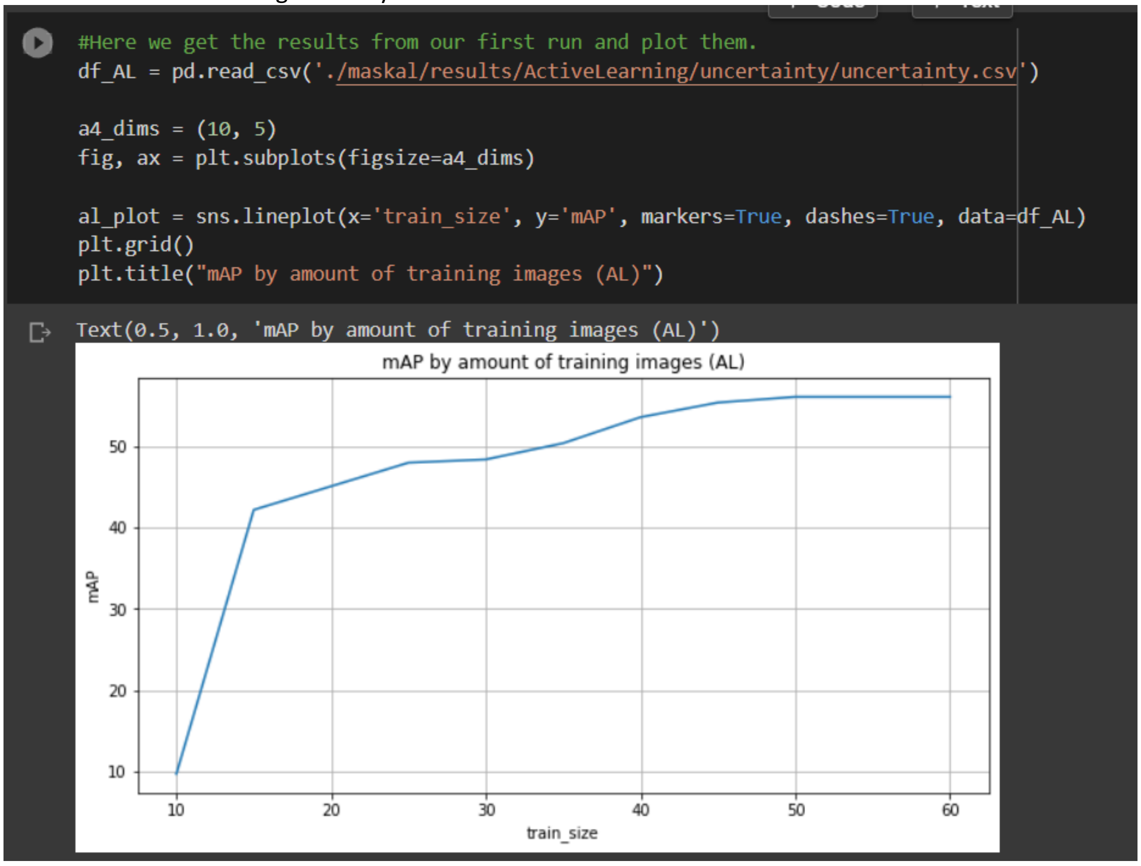
```
Labels
  2937599387_80e7d6e050_b.xml
  4543126482_92254ef046_b.xml
  4887227769_acd2e6127d_b.xml
  5178670692_63a4365c9c_b.xml
  7178882742_f090f3ce56_k.xml
```

5. Now, upload the labels to the 'annotate' folder under: maskal/AL-demo-files/train/annotate



```
maskal
  AL-demo-files
    test
    train
      annotate
        10464445726_6f1e3b...
        12037308314_e16fb3...
```

6. MaskAL will detect the labels and start training and evaluating again.
7. The loop will start back at step 2, 10 more times but with 5 images at a time.
8. Once the training is done you can visualize the results.

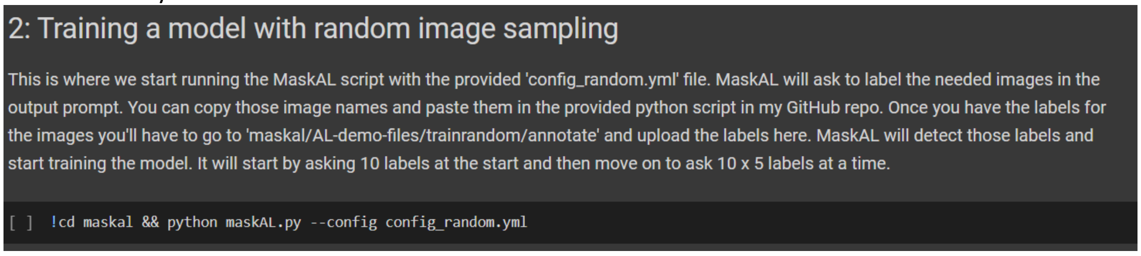


```
#Here we get the results from our first run and plot them.
df_AL = pd.read_csv('./maskal/results/ActiveLearning/uncertainty/uncertainty.csv')

a4_dims = (10, 5)
fig, ax = plt.subplots(figsize=a4_dims)

al_plot = sns.lineplot(x='train_size', y='mAP', markers=True, dashes=True, data=df_AL)
plt.grid()
plt.title("mAP by amount of training images (AL)")
```

```
Text(0.5, 1.0, 'mAP by amount of training images (AL)')
```

Running the random image sampling script

1. Now you can move on to the second run.

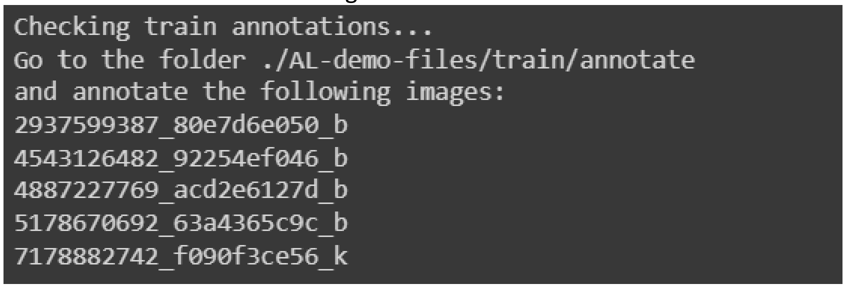


2: Training a model with random image sampling

This is where we start running the MaskAL script with the provided 'config_random.yml' file. MaskAL will ask to label the needed images in the output prompt. You can copy those image names and paste them in the provided python script in my GitHub repo. Once you have the labels for the images you'll have to go to 'maskal/AL-demo-files/trainrandom/annotate' and upload the labels here. MaskAL will detect those labels and start training the model. It will start by asking 10 labels at the start and then move on to ask 10 x 5 labels at a time.

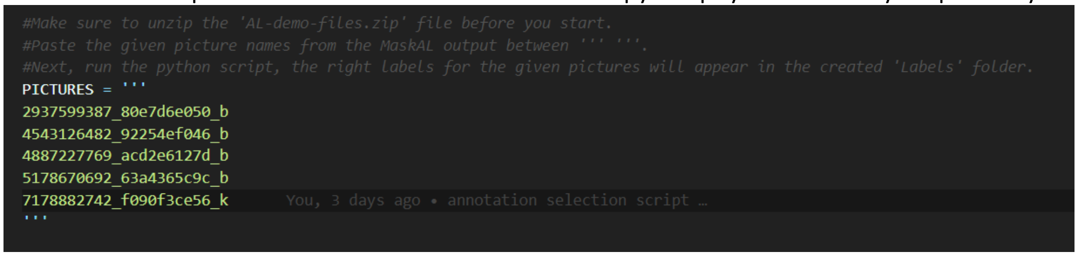
```
[ ] !cd maskal && python maskAL.py --config config_random.yml
```

2. The MaskAL script will give you a list of images that need to be labelled. The very first time it will ask to label 10 images.



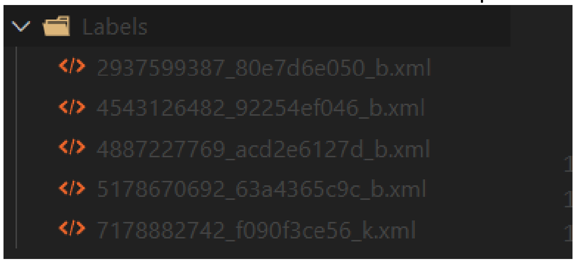
```
Checking train annotations...
Go to the folder ./AL-demo-files/train/annotate
and annotate the following images:
2937599387_80e7d6e050_b
4543126482_92254ef046_b
4887227769_acd2e6127d_b
5178670692_63a4365c9c_b
7178882742_f090f3ce56_k
```

3. You can paste this list in the 'select-annotations.py' script you cloned to your pc locally earlier.



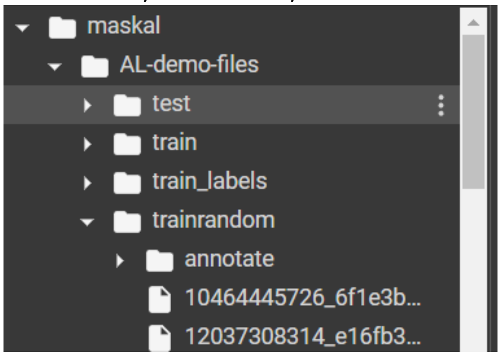
```
#Make sure to unzip the 'AL-demo-files.zip' file before you start.
#Paste the given picture names from the MaskAL output between ''' '''.
#Next, run the python script, the right labels for the given pictures will appear in the created 'Labels' folder.
PICTURES = '''
2937599387_80e7d6e050_b
4543126482_92254ef046_b
4887227769_acd2e6127d_b
5178670692_63a4365c9c_b
7178882742_f090f3ce56_k        You, 3 days ago • annotation selection script …
'''
```

4. Once the image names are pasted between the "' "', you can run the python script. It will create a 'Labels' folder with the requested labels in it.

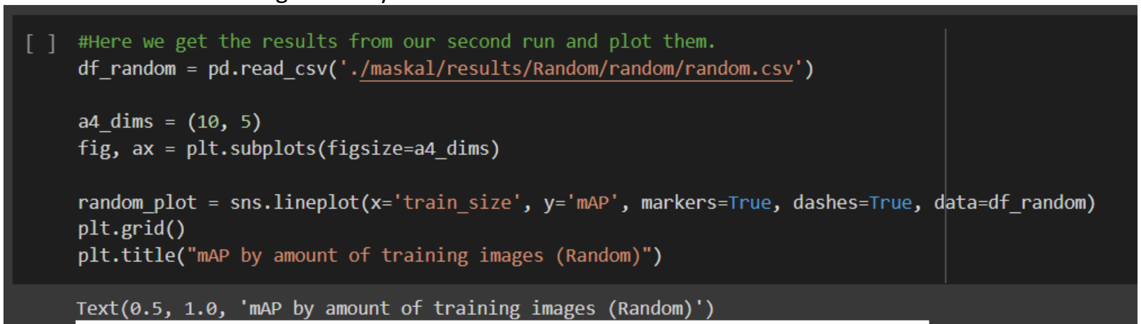


```
Labels
  2937599387_80e7d6e050_b.xml
  4543126482_92254ef046_b.xml
  4887227769_acd2e6127d_b.xml
  5178670692_63a4365c9c_b.xml
  7178882742_f090f3ce56_k.xml
```

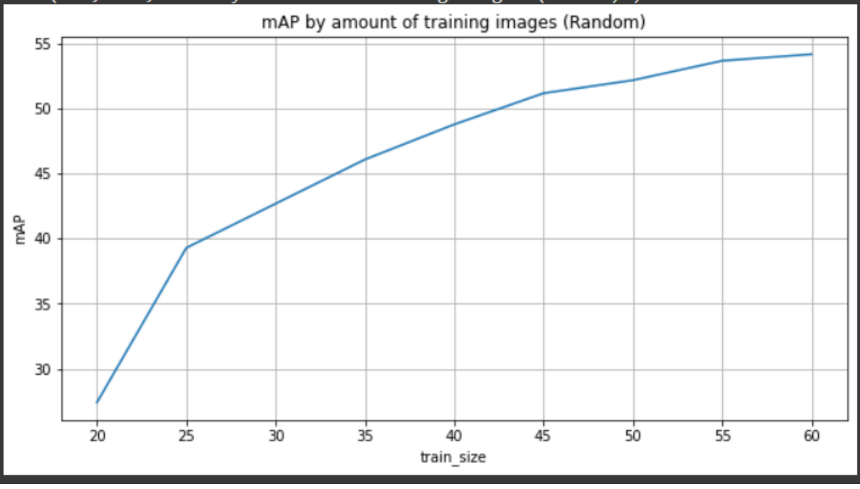
5. Now, upload the labels to the 'annotate' folder under: maskal/AL-demo-files/trainrandom/annotate



6. MaskAL will detect the labels and start training and evaluating again.
7. The loop will start back at step 2, 10 more times but with 5 images at a time.
8. Once the training is done you can visualize the results.

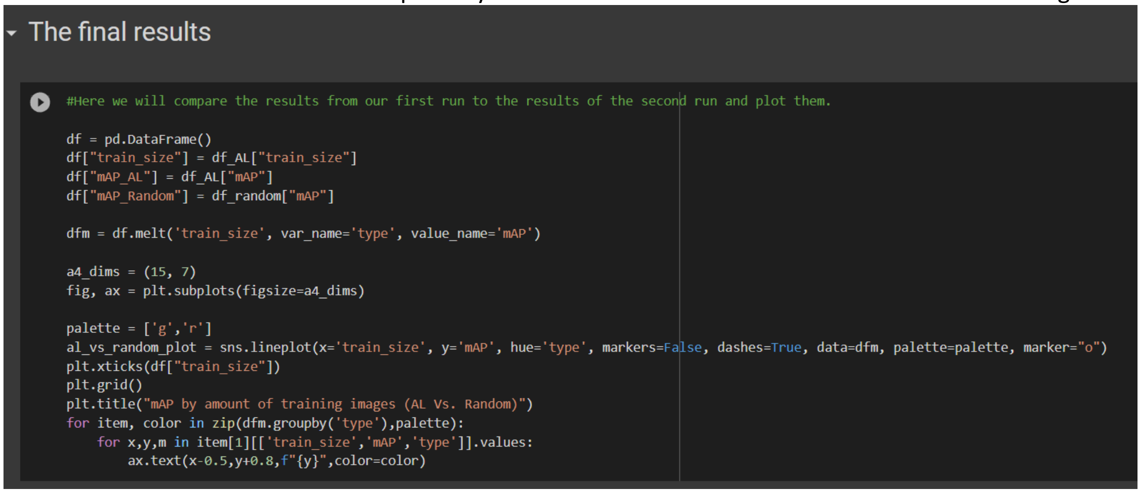


```
[ ] #Here we get the results from our second run and plot them.
    df_random = pd.read_csv('./maskal/results/Random/random/random.csv')

    a4_dims = (10, 5)
    fig, ax = plt.subplots(figsize=a4_dims)

    random_plot = sns.lineplot(x='train_size', y='mAP', markers=True, dashes=True, data=df_random)
    plt.grid()
    plt.title("mAP by amount of training images (Random)")

    Text(0.5, 1.0, 'mAP by amount of training images (Random)')
```



Visualizing final results

1. Once the second run is completed you can visualize the final results and run the following cell.

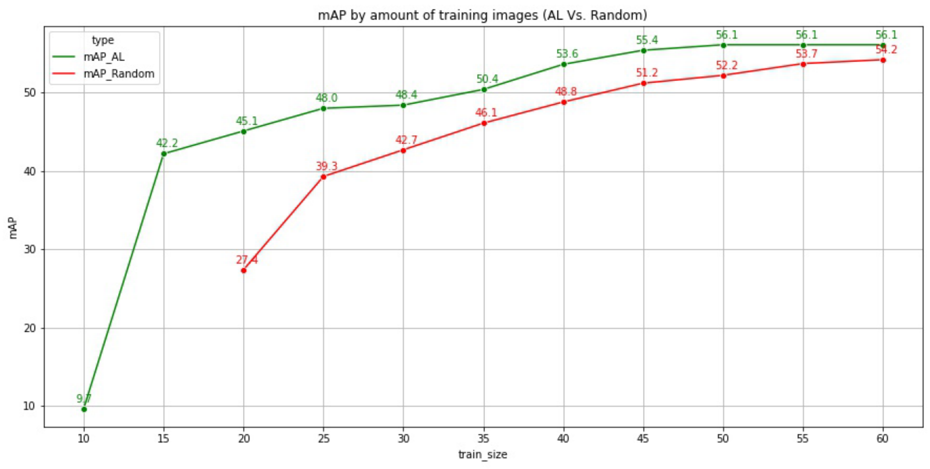


The final results

```
#Here we will compare the results from our first run to the results of the second run and plot them.

df = pd.DataFrame()
df["train_size"] = df_AL["train_size"]
df["mAP_AL"] = df_AL["mAP"]
df["mAP_Random"] = df_random["mAP"]

dfm = df.melt('train_size', var_name='type', value_name='mAP')

a4_dims = (15, 7)
fig, ax = plt.subplots(figsize=a4_dims)

palette = ['g','r']
al_vs_random_plot = sns.lineplot(x='train_size', y='mAP', hue='type', markers=False, dashes=True, data=dfm, palette=palette, marker="o")
plt.xticks(df["train_size"])
plt.grid()
plt.title("mAP by amount of training images (AL Vs. Random)")
for item, color in zip(dfm.groupby('type'),palette):
    for x,y,m in item[1][['train_size','mAP','type']].values:
        ax.text(x-0.5,y+0.8,f"{y}",color=color)
```

Interpreting results

You'll get a graph that should look similar to this one.

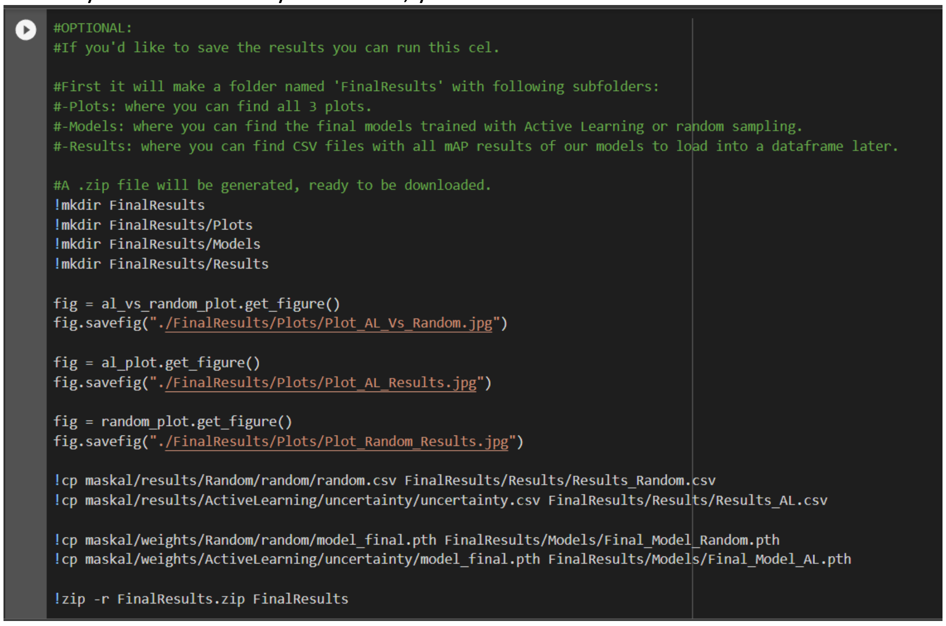


The green line are the results from our first run with Active Learning; this is where the input pictures where selected by MaskAL.
The red line are the results of our second run with random sampling; this is where the input images are completely randomly selected.
x-axis: the amount of training images the model was trained on
y-axis: the mean average precision of the model

In this example, our model trained with active learning already had a better result with only 75% of our training data and had maximum results with 83,3% of our total training data.

Saving your results

In case you want to save your results, you can run the final cell.



```
#OPTIONAL:
#If you'd like to save the results you can run this cel.

#First it will make a folder named 'FinalResults' with following subfolders:
#-Plots: where you can find all 3 plots.
#-Models: where you can find the final models trained with Active Learning or random sampling.
#-Results: where you can find CSV files with all mAP results of our models to load into a dataframe later.

#A .zip file will be generated, ready to be downloaded.
!mkdir FinalResults
!mkdir FinalResults/Plots
!mkdir FinalResults/Models
!mkdir FinalResults/Results

fig = al_vs_random_plot.get_figure()
fig.savefig("./FinalResults/Plots/Plot_AL_Vs_Random.jpg")

fig = al_plot.get_figure()
fig.savefig("./FinalResults/Plots/Plot_AL_Results.jpg")

fig = random_plot.get_figure()
fig.savefig("./FinalResults/Plots/Plot_Random_Results.jpg")

!cp maskal/results/Random/random/random.csv FinalResults/Results/Results_Random.csv
!cp maskal/results/ActiveLearning/uncertainty/uncertainty.csv FinalResults/Results/Results_AL.csv

!cp maskal/weights/Random/random/model_final.pth FinalResults/Models/Final_Model_Random.pth
!cp maskal/weights/ActiveLearning/uncertainty/model_final.pth FinalResults/Models/Final_Model_AL.pth

!zip -r FinalResults.zip FinalResults
```

It will create a 'FinalResults.zip' file you can download. In the 'FinalResults' folder you'll find:

- The 2 generated graphs.
- The 2 final AI models.
- The 2 CSV files with each model's mAP values.

howest
hogeschool

howest.be